**Diário de bordo – 23.02.2024**

Article.aspx@id=333

O ficheiro python *get.artigos* que fizemos na aula anterior está a produzir documentos com base nos muitos documentos *'article'*.

O que queremos agora é algo que, a partir desses documentos, crie outro documento para o nosso arquivo/dataset. Para isso temos de ver o que limpar e definir o que produzir. Não faz sentido limpar se não tivermos uma direção onde confluir.

*O que gostávamos que fosse a saída?*

*O que vamos fazer com os documentos se não tivermos em conta o tipo de uso que vamos ter com eles (ter o formato certo, o que é para manter ou deitar fora, etc)?*

**Primeiro temos de fazer trabalho de planeamento!**

Já falamos que vamos ter de produzir alguma espécie de corpus onde possa ser feita as análises de quantos documentos contêm, que frequência de palavras, que entidades, verbos, etc

**Ter um corpus/artigo acerca da UM e acerca dos jornais da UM e de futuras entrevistas.**

Para conseguir isto temos interesse em ter associado a cada documento algum tipo de **METADADOS**. Precisamos de ter um formato que nos dissesse como explicito os metadados (o que é um parágrafo, etc).

Formatos com que podemos pegar:

- **word** (o complicado de processar word é que por omissão é um formato binário o que torna difícil procurar palavras e ver diferenças);
- **markdown** (tem algo bom que até é elegível, tem um sítio natural para colocar metadados e deixar texto corrido, para marcar parágrafo podemos apenas deixar uma linha em branco, permite criar cabeçalhos de secção sobre secção, não tem dificuldade em colocar uns itemize e coloca imagens).

**O FORMATO NORMAL DO MARKDOWN:**

- - -

**Metadados:** *yaml*, atributos e valores.

A partir daqui pomos os outros dados que nos lembremos.

Por exemplo, *title: título* ; *autor: nome do autor* ; *type: o tipo de documento que temos* ; *about: campus de Gualtar, bibliotecas*.

Se não soubermos algum metadado podemos mais tarde editá-lo à mão, se não soubermos mesmo podemos ignorar.

- - -

**Markdown:** elemento, linha em branco, elemento, linha em branco. Ao ter um título importante: *título*. Um título menos importante: **subtítulo**.

O que queremos também: que apareçam todos os documentos das nossas entrevistas e incluir entrevistas do âmbito do 'Museu da Pessoa'.

Sugestão: guardar os articles, manter o original para que se virmos que há mais coisas que temos de limpar é mais fácil fazê-lo olhando para o original.

Daqui a uma semana: ter uma versão inteira junta dos artigos todos. A parte textual dos artigos não é pesada, mas ao juntar o que descarregarmos sim. Ter alguma política.

Assim o que temos de fazer...

1. Ver o que limpar (criar uma lista)
2. Como transformar um formato que não é o ideal noutro formato
3. Dividir o problema grande em problemas mais pequenos

**PASSANDO AO CÓDIGO:**

Vamos trabalhar com os articles 1139-1146.

*Nota: fazer limpeza de metadados*

Ir à saída que criamos com o código – localizar – palavra 'entrevista' – ver quais artigos são entrevistas. Vamos analisar e trabalhar com o **article id=3302** que possui entrevista.

Num *get.artigos* (código do Maurício):

Temos um *proc_article (process article)* que processa um ficheiro html article.aspx@id=000 do jornal NosUMinho.

A linha 6 vai buscar os artigos todos. Depois do aspx inserir ***** (que significa qualquer sequencia de caracteres). Estamos a dizer especificamente o artigo 3302.

Da linha 28-35 está a pensar em metadados.

Vamos criar um *proc_artcontents (process article content)* onde nos vamos concentrar em apenas 1 dos artigos.

Na linha 36: *art = a.find("div", id="artigo")* – estamos a dizer que o artigo é obtido à custa de ir buscar uma *div* que tem o identificador de ser arquivo e fizemos um *find* que o vai buscar diretamente.

37: *corpo = proc_artcontents*

44: *tag.extract()* extraímos as tags e todos os 'voltar' são apagados

Processa o artigo e escreve o corpo, como lhe demos uma árvore documental vai escrever com as etiquetas todas.

Assim, ao processar aparece uma **saída com todas as tags**.

*Nota: ter à mão um sítio onde procurar documentação especifica (por exemplo, o beautiful soup que estamos a usar).*

*Erro name 'glob' is not defined: podemos importar no jjcli e não fazer import glob*

45: *tag.decompose()* vamos deitar fora o *div* que diz *slidesjs-log*

46: tudo o que tenha a etiqueta de classe a incluir *social_count* vai apagar

47: definimos para todas as ocorrências do *id= "slides"*

**.extract:** apaga da árvore, mas guarda.

**.decompose:** 'varre a casa e põe o lixo no contentor'. Elimina mesmo da árvore.

**E tabelas...**

Quando aparecer a etiqueta <tabela> queremos que apareça ## TABELA, juntando a string e processando o que está lá dentro.

50-51: *tag.insert(0, "\n ##TABELA")* – vamos procurar todas as tabelas e juntar aos nossos elementos.

52: *finalt (texto final) = ar.get_text()* – dá-nos parágrafos já com linhas em branco

53: *finalt = re.sub(r"\n{3, }", r"\n\n", finalt)* – estamos a dizer para substituir uma coisa por outra, nas tabelas fez limpeza de linhas, deixamos de ter algumas que estavam a mais, fica mais organizado.

**TPC:**

1. Dissecar a Biblioteca Digital (qual a estrutura e que tipo de documentos tem)
2. Definir o nosso kit de metadados (título, data, autor, ...)
3. Procurar as etiquetas  e mudar para <b>
4. Criar uma função que processe slides e que escreva noutro ficheiro o nome do jpg e da caption
5. Ir buscar os créditos de <h1> e extrair para os metadados
6. Inserir os - - - e os * para os títulos